X-601-72-7

PREPRINT

STANDARDS GUIDE FOR SPACE AND EARTH SCIENCES COMPUTER SOFTWARE

JANUARY 1972

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-601-72-7

STANDARDS GUIDE

FOR

SPACE AND EARTH SCIENCES

COMPUTER SOFTWARE

January 1972

FOREWORD

This document was prepared by a subcommittee of the Space and Earth Sciences Computer Users' Committee with the following membership:

Mr. G. Mason (Chairman)	Code 626
Dr. R. Chapman	Code 682
Dr. D. Klinglesmith	Code 671
Mr. J. Linnekin	Code 601
Mr. W. Putney	Code 603
Mr. F. Shaffer	Code 603
Mr. R. Dapice (Consultant)	Code 266

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION AND HISTORY

The Space and Earth Sciences Directorate (SESD) Computer Users' Committee has been concerned for a long time with insuring the most efficient use of the available resources by its constituents. On September 10, 1970, a subcommittee was formed to develop a set of guidelines for the preparation of Systems Analysis and Programming work statements. The original intent was to use the guidelines for contract work, but they can serve equally well for in-house efforts. This document is a consolidation of data from many sources but is not in any way exhaustive in its detail. The use of these guidelines in whole or in part is completely voluntary, but we feel that the results obtained through their use could contribute greatly to the efficient administration of available monetary and equipment resources.

Throughout the remainder of the document, you may find ideas that can be used in their entirety or in part for a given task. Seldom will they all be useful in any single instance, but, used where appropriate, they may make your part of the project easier. Emphasis is placed on language standards and the application of good management techniques to software development.

2. MANAGEMENT

The work statement can provide several important management tools for the Technical Officer (T.O.). Some of the most useful of these are the milestone and staffing charts, the design document, and technical meetings. The milestone chart (Figure 1) is used to specify all points in the life of the project which require a sponsor/contractor interface. The interface may be in the form of a meeting, a document, or a signature of approval. Items on the chart are listed in order of anticipated occurrence. When making an estimate of personnel requirements for the task, you can assign "time to completion" figures to each milestone. These numbers can be compared with those supplied by the contractor when submitting his proposal. The major benefits of a milestone chart are that it enables you to compare the contractor's estimates with your own and that it provides a firm list of the accomplishments leading to successful completion of the project.

The staffing chart is similar to the milestone chart. It is used to show what each person assigned to the task will be expected to do during his period of involvement. The major labor milestones are plotted along a time scale, and personnel assignments are shown, one line per person (Figure 2). In addition to these forms, 11 other forms are delineated in Section 4 of this report. These are designed to allow for an orderly and manageable software development process for a large system.

In a project of any appreciable size that includes a design effort, one important requirement is the design document. This document should be prepared by the contractor and should embody the results of the design portion of the project. The contents should reflect the original statement of work and the developments that occurred to modify it, if any. This document should be reviewed by the T.O. and either approved or returned with comments on required improvements or changes. Generally, the creation of this document occurs after a series of dialogues between both sides, and it is, therefore, usually close to the desired result when first submitted. After approval, which should be a predetermined milestone, no significant changes should be permitted without prior written approval of the T.O. This procedure helps to insure an orderly passage from design to implementation and allows documentation of any deviations that might affect project cost or schedule.

The primary mode of communication between the contractor and T.O. is the periodic meeting. No established rule exists for setting the frequency, but weekly meetings, at least until delivery of the design document, are usually reasonable. No matter what the schedule, the opportunity should exist for meetings or phone calls whenever a situation arises that

Programming Schedule	Date Completed
Functional Specifications*	
Design Specifications*	
Flow Charts*	
Coding	
Checkout	
Acceptance Tests*	
Format Delivery of Programs	
Documentation Schedule	
Outline*	
First Draft*	
Initial Artwork*	
Edited Draft*	
Camera Ready Artwork*	
Formal Delivery of Documentation	

Figure 1. Milestone Chart**

*Government review and approval required before work on next phase begins.

**Entries shown here are minimum. Others may be added to reflect requirements of individual tasks.

STAFFING CHART

TASK # _____ DATE _____

TASK NAME _____

T.O. _____

SCALE: 1 DIV= _____

MILESTONE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
FUNCTIONAL SPECS																
DESIGN SPECS																
FLOW CHARTS																
CODING																
CHECKOUT																
ACCEPTANCE TESTS																
FINAL DELIVERY																

Figure 2. Staffing Chart

Milestones used on this type of chart can be at a much lower level than shown on the milestone chart to maintain closer control over efforts expended.

requires rapid resolution. All meetings and calls should be documented. Both parties should keep notes, which should document new developments and action items decided upon. All action items should show whose action is required and when delivery is expected.

When problems are encountered that might affect the schedule, the contractor should provide full documentation that explicitly defines what the problem is, what solution (if any) has been found, and the time and money losses that might be incurred. Failure to provide this documentation can severely impair the schedule and cause great difficulty in recovering the schedule at a later time.

Generally, the task originator has an idea of the particular programming language he wishes to be used in the task. This choice is based on several factors, including language capability and ease of program maintenance. The choice, once made, should be adhered to stringently. Deviations should require approval by the Technical Officer. Of prime importance are those areas which interface with the operating system and, as a result, might be subject to difficulty when a new version of the operating system is installed. If the choice of a language is not clear-cut, you could request proposals for more than one or have the contractor propose a language based on his determination of what the proper language should be. Since each language has its advantages and its liabilities, allow yourself to obtain the best product by implementing your programs in the best one for your task. A summary of user software available at GSFC is contained in the GSFC Computer Program Library Catalog. Principally, this software consists of general purpose closed routines to perform mechanical functions (print pagination, acquisition of system data, etc.) or mathematical ones (coordinate conversion, table look-up, etc.). Often, some of this software will satisfy part of the task requirements. When this situation arises, the contractor should be advised of its availability. If the contractor elects to use the software, however, he should accept the responsibility for its proper operation. His choice is one of design and write versus test and use. Should you insist on the use of some of these programs, you will probably have to accept the responsibility for their integrity. In general, the less maintenance you provide the better; hence, attempt to have all portions of the software be the responsibility of the contractor.

During the task, the contractor should be required to maintain adequate back-ups of source programs, compiled programs, and test data sets. Source programs can be maintained on tape efficiently on the IBM 360 with the program SLIP. This program allows the selective updating of individual modules and the retention of different versions of the source program as well as data decks. Most manufacturers supply similar utility programs as well as programs for preparing copies of other types of data sets.

The contractor should acquaint himself with system availability, operating restrictions, hardware resources, and all related data concerning the system he is working on. Items 1 to 10 in the bibliography are available to all personnel and should be used as a starting point for locating all required information. Because the computer situation is a very dynamic one, operating restrictions and system resources change. Up-to-date, detailed information is available at the Programmer Assistance Centers located around the Center. These groups also serve as focal points for problems encountered with the hardware and software systems.

3. PROGRAMMING STANDARDS

3.1 GENERAL

Programming standards for the SESD have been designed to assist the programmers in writing programs that are compatible. The standards will also further ease the modification to the programs and aid in the management control of the software systems developed for the SESD.

3.1.1 Language Specification

One of the most difficult problems that has plagued users of high-speed digital computers is the use of programming techniques that fall outside of the language specification. For this reason we recommend that only those language elements defined in USAI Standard FORTRAN (Ref. 1) supplemented by the following vendor dependent (IBM 360) syntax be used for FORTRAN program development on SESD 360 computers.

Direct Access Input/Output Statements
Double Exponentiation
END and ERR parameters in READ
ENTRY
Generalized Subscripts
Hexadecimal Constant
IMPLICIT
Initial Data Values in Type Statement
Length of Variables as Part of Type Specifications
Literal Enclosed in Apostrophes
Mixed Mode Expressions
More Than Three Dimensions in an Array
PAUSE 'message'
PRINT
PUNCH
READ b, list
T and Z Format Codes
RETURN i

3.1.2 Other Guidelines

The guidelines that are delineated in Sections 3, 4, and 5 of this report were developed under contract NAS5-11723 (Ref. 2) for Dr. C. E. Velez of Code 553.1, GSFC. They have been modified where necessary to fit the requirements of the SESD, but much of the material appears intact.

In the sections that follow, an effort is made to provide guidelines for the development of a large software system such as that required for the flight projects associated with the SESD. For project related or other large software ventures, it is recommended that all of the control procedures including the project library be used. For smaller development efforts, the standards should be used, but the degree of other control will be left to the discretion of the Technical Officer.

3.2 LANGUAGE

The following standards are provided for FORTRAN-IV, H, the most widely used programming language in the Directorate. The object modules used in system integration are to be compiled with the level H compiler, optimization level of 2.

3.3 CORE STORAGE

The core storage required for any one job step will be limited as specified in Section 7. Unlabeled COMMON will be used as "scratch pad" temporary storage so that total storage requirements for the program will be reduced.

3.4 NAMING CONVENTIONS FOR VARIABLES

a. A standard naming convention for variables will be followed in all cases. Integer variables will use names that start with one of the letters I, J, K, L, M, or N. Real variables will be assigned names starting with letters A-H and P-Z. Logical variables will have names starting with the letter O. Zeros will never be used in variable names.

b. Subroutine names should suggest the function performed within the routine.

c. Standard names will be established for all frequently used (major) variables which may be used within several subroutines, and these names will be used throughout the program. A current glossary of such names will be maintained in the project library.

3.5 SUBROUTINE COMMUNICATION

Unlabeled COMMON is to be used as temporary storage within subroutines and may be used additionally as the communication medium for large blocks of arguments from one subroutine to another where there is only one transfer level. The integrity of this scratch pad memory is maintained only from the calling program to the called program or in the direct return process.

The number of arguments in a subroutine CALL argument list will be minimized (no more than eight). Whenever possible, arguments will be passed by value rather than by name when required in subroutine CALL lists. The total number of labeled COMMON statements allowed within any subroutine will be restricted to 12. Variables passed to a subroutine through COMMON should be identified by the use of comment cards or by the EQUIVALENCE statement. For example, in a subroutine, the statements

```
COMMON/ALPHA/DUM(50)
EQUIVALENCE (DUM(4),AE)
```

identify the use of the variable DUM(4), designated in the subroutine as AE, from the ALPHA COMMON area.

Labeled COMMON will be formatted to have slashes in columns 13 and 20 and have five or less variables per line starting with column 22, spaced 10 columns apart, with commas in columns 32, 42, 52, 62, and 72 as required. For example:

```
Col. No. 7    13        20   22    33    43     53        63          72
         COMMON/    ALPHA/   AE    ,GM   ,WE    ,RADINA   ,X(20,3)    ,
Col. No. 6                   22    33    43
         *                   XD    ,XOD  ,XXDD
```

A similar convention (starting in column 22) will be used for all DATA, TYPE, DIMENSION, and EQUIVALENCE statements. For example:

```
Col. No. 7          22       33         43
         DIMENSION  A(2)     ,GN(2)     ,X(20,3)
```

The COMMON statement is to contain all the dimension information for its variables. Each COMMON statement is to refer to one named COMMON area only. No named COMMON area should appear in more than one COMMON statement unless too many entries exist to allow it. In that case, multiple statements must be contiguous.

3.6 COMMENT CARDS

Each subroutine (see page 3-5) will have sufficient comment cards following the subroutine name to:

a. Identify the function performed by the subroutine
b. Define or reference the mathematical model
c. Explicitly define each of the arguments in the CALL list
d. Identify the major variables
e. Explain labeled COMMON
f. Identify the subroutines used
g. Explicitly identify the input and output

Comments will be of sufficient length to enable other programmers to follow the flow and determine the purpose of the various parts of the program. Comments will be added to the coding as it is generated while the reasons for steps are still obvious to the programmer.

The first card of each function or subroutine will be labeled FUNCTION or SUBROUTINE. (Comments will not be placed prior to the subprogram definition.) Sufficient comment cards will be in each subroutine at each major switching statement (IF, GO TO, CALL, etc.) to provide a readable logical flow of the program.

Each subroutine will have comment cards to show the programmer's name and the date the subroutine was completed. When program modifications are made, the date and programmer's name will be added.

A comment card or series of comments may be preceded by a "blank" comment card to create the illusion of "paragraphing" the code. Where feasible, two blank cards preceding and one following a long comment present a neat appearance. Refer to the subroutine listing shown in Figure 3 for an example of the proper use of comment cards following the subroutine name.

3.7 INPUT/OUTPUT

All program input/output will be performed within a minimum number of subroutines which will be called from the executive program. All card input is to be performed in a single routine. Input card images should be printed out as they are read, allowing for carriage control as required. All READ statements should use "error" and "end" exits where possible.

```
      SUBROUTINE MOONAD(DJ,ET,SE,CE,RA,DEC,R)
C     VERSION OF 10/28/63
C        FORTRAN SUBROUTINE FOR FORTRAN 2 MONITOR ON IBM 7090, 7094
C     PURPOSE
C        COMPUTES APPARENT RIGHT ASCENSION, DECLINATION, AND
C        HORIZONTAL PARALLAX OF THE MOON
C     CALLING SEQUENCE
C        CALL MOONAD (DJ,ET,SE,CE,RA,DEC,R)
C     INPUT
C        DJ  = JULIAN DATE AT 0 HOURS EPHEMERIS TIME
C        ET  = EPHEMERIS TIME IN RADIANS (24 HOURS = 2 PI RADIANS)
C              ET IS RESTRICTED TO LIE BETWEEN 0 AND +2 PI RADIANS
C        SE  = SINE TRUE OBLIQUITY OF DATE
C        CE  = COSINE TRUE OBLIQUITY OF DATE
C              NOTE - REFER TO FUNCTION EQN TO COMPUTE TRUE
C              OBLIQUITY OF DATE
C     OUTPUT
C        APPARENT COORDINATES OF THE MOON
C        RA  = APPARENT RIGHT ASCENSION IN RADIANS
C        DEC = APPARENT DECLINATION IN RADIANS
C        R   = HORIZONTAL PARALLAX IN SECONDS OF ARC
C     REFERENCE
C        "ASTRONOMICAL PAPERS PREPARED FOR THE USE OF THE AMERICAN
C        EPHEMERIS AND NAUTICAL ALMANAC" VOLUME 15, PART 1
C        ("THEORY OF THE ROTATION OF THE EARTH AROUND ITS CENTER
C        OF MASS" BY EDGAR W. WOOLARD - PAGES 53, 64-66)
C     METHOD
C        WOOLARD HAS FORMED AN ABBREVIATED VERSION OF BROWN'S
C        THEORY OF THE MOON IN HIS PAPER MENTIONED IN REFERENCE.
C     REQUIRED SUBPROGRAMS - FORTRAN 2 MONITOR
C     ANALYSIS
C        RICHARD J. SANDIFER, DATA SYSTEMS DIVISION
C                             GODDARD SPACE FLIGHT CENTER, NASA
C     PROGRAMMER
C        SHIRLEY G. STATEN, DATA SYSTEMS DIVISION
C                           GODDARD SPACE FLIGHT CENTER, NASA
C     PROGRAM MODIFICATIONS
C        1/15/63 ORIGINAL SUBROUTINE MOON BY SHIRLEY STATEN
C        10/28/63 MODIFICATION BY J. PEACOCK, DATA SYSTEMS DIVISION
C                 NAME CHANGED TO MOONAD, OUTPUT GIVES RIGHT ASC.
C                 AND DEC. INSTEAD OF LONG. AND LAT. CALLING SEQ.
C                 ENLARGED TO INCLUDE SIN AND COS OF OBL.
C*******START PROGRAM**************************************************
C
```

Figure 3. Example of Comment Cards Following Subroutine Name

The logical unit for all data sets will be a variable name but not variable at run time. A variable number for the unit will be assigned by a DATA statement. All output data sets will be blocked at a reasonable number of logical records per physical record (such as, BLKSIZE=7265, LRECL=137 for printed output, and LRECL=80, BLKSIZE=7280 for source code data sets.)

3.8 CONSTANTS

a. There will be a labeled COMMON area for absolute constants and a separate labeled COMMON area for physical constants that have nominal values. These and other initial values will be set by DATA statements with the BLOCK DATA subprogram.

b. All REAL*8 constants must have a 'D' form exponent appended to them.

3.9 STATEMENT LABELS

Statement labels are to be right adjusted. The format labels are as follows:

a. All input format statement labels will be numbered 1000-1999.

b. All output format statement labels will be numbered 2000-2999.

All subroutine RETURN labels are to be 999. Executable statement labels will be in numerical order where possible and labeled 1-998. To allow for subroutine modifications, statement numbers should be initially incremented by 10.

To facilitate the use of debugging tools, label numbers will be assigned to all GO TO and CALL statements.

3.10 ERROR CONDITIONS

Under no condition will a subroutine return control to the S/360 Operating System; i.e., the STOP statement will not be used. An error condition code will be established which will be standard for the entire system. All error codes from a subroutine will be returned to the calling program for appropriate action.

3.11 MISCELLANEOUS

a. All DO loops containing two or more statements are to end with a labeled CONTINUE statement.

b. Parentheses add readability to FORTRAN equations and should be used to separate computation rather than rely on the FORTRAN hierarchy of operators.

c. The length of an individual subroutine will be limited to 150 executable source statements plus comment cards.

d. To increase readability, blanks should be used to separate the variable names from the operators in complex statements.

e. The necessary specification statements in each subroutine will be arranged in the following order at the beginning of the program: (No comments will separate this information although they may precede it.)

   (1) IMPLICIT Statements
   (2) TYPE Statements
   (3) COMMON BLOCKS
   (4) DIMENSION Statements
   (5) EQUIVALENCE Statements
   (6) DATA Statements

f. Mixed mode arithmetic statements should be avoided.

g. Subroutine names should not be passed through argument lists.

h. Multiple evaluation of functions with the same argument within a subroutine must be avoided.

i. Use assigned GO TO statements rather than multiple IF statements.

j. Debug runs should be limited to maximum of 1 minute.

k. Assembly language lists and SYSUDUMP and SYSABEND core dumps for FORTRAN programs are not to be generated unnecessarily.

l. Documentation is to be maintained at a current level by each programmer.

m. All format statements will be placed at the end of a subroutine, and all printed messages will have the first 13 characters as follows:

PPPPPP NNNN

where PPPPPP is the routine name, left adjusted and padded on the right with blanks, and NNNN is the format statement number.

The suggestions below can be followed to optimize FORTRAN code, to achieve better accuracy, and to increase speed of execution. Note, however, that statements introduced to achieve better accuracy and speed can reduce the clarity of your code and make debugging more difficult.

n. Apply the information gained during analysis of your problem before programming to simplify the problem and speed up the numerical procedure.

o. Arrange the program logic to avoid branches whenever possible.

p. Make the most probable result of all logical IF statements a simple drop-through instead of a branch.

q. Reduce I/O to the minimum necessary when debugging is complete.

r. Use implied DO loops in input/output in place of I/O within actual DO loops whenever possible.

s. At the beginning of the program, calculate all quantities that are constant throughout the program and calculate all quantities constant throughout a loop outside the loop.

t. Use as few subscripts as possible on arrays; for example, A(720) instead of A(12,6,10).

u. Using IF statements to determine conditional branches to more than three labels is less efficient than using unconditional or computed GO TOs. The computed GO TO uses more overhead time and space than the unconditional GO TO, but it is still more efficient than a set of IF statements.

v. Whenever possible, pass variables to subroutines through COMMON instead of using parameter lists.

w. Because of round-off error in low-order bits, do not test for equal using floating point variables. Use .GE. or .LE.

x. Use SQRT instead of **.5. For small powers, use A*A*A* ... or A**I with I=R instead of A**R, where R is a floating point integer. (Values raised to integer powers are computed by repetitive multiplication, whereas values raised to real powers are computed using logarithm and exponential routines. A**R is about four times slower than A**I.)

y. Use unformatted I/O for data files whenever possible.

z. Perform all initializations in one routine.

aa. Where programs are expected to use large amounts of computer time, facilities should be built in to allow restarting of the program at frequent intervals.

3.12 OPTIMIZATION

Program optimization can occur at several times during a project. The first, and most useful, is during the design stages. There is no substitute for careful consideration of resource use prior to program implementation. During development, careful programming and use of optimizing language processors can enhance the efficiency of a program (see Sections 3.1 to 3.11). After a program has been written, several means are available for measuring and improving efficiency. Among these are the Boole and Babbage SMS package and the services of the Programmer Assistance Centers. Users are urged to make maximum use of these facilities in order to reduce the burden to themselves and to the entire user community by eliminating wasteful computer practices.

4. DOCUMENTATION STANDARDS

4.1 GENERAL

This section defines the SESD documentation requirements which should be used for large software development efforts. These requirements include a Project Library, Flowchart Symbols, Task Specifications, a Subsystem Design Manual, a User's Manual, and Project Standard Forms. In addition to the Documentation Standards described here, a document published recently by NASA Headquarters, entitled "Computer Program Documentation Guideline" (Ref. 3), contains some general guidelines for program documentation.

4.2 PROJECT LIBRARY

A project library will be established under the control of a librarian. The library will contain:

a. Source Code Decks
b. Object Code Decks
c. Program Listings
d. Program Documentation
e. Standard Error Codes
f. Test Reports and Results

In addition, the librarian will be responsible for the maintenance of the following:

a. Task Assignment Form
b. Subroutine Assignment Form
c. Parameter Glossary
d. Project Subroutine Control List
e. Project Documentation Control List

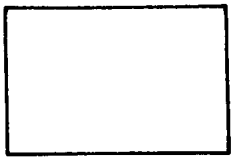
4.3 FLOWCHARTING SYMBOLS

All source programs should be run through AUTOFLOW, and, when it is necessary to flowchart by hand, the following symbols should be used:

ENTRY/EXIT

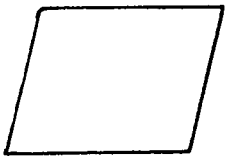
This symbol represents an entry to or exit from a subroutine.

PROCESSING



This symbol represents a group of instructions which perform a processing function of the subroutine.

INPUT/OUTPUT

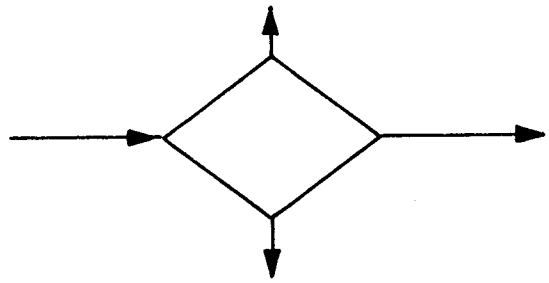


This symbol represents an input/output (I/O) process. (It should be used when the I/O unit or device can vary.)

FLOW DIRECTION

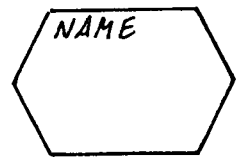
This symbol represents the direction of processing of data flow. Normal direction of flow is left to right. Arrowheads are necessary on all lines of flow. If it is necessary to use lines which oppose the normal direction of flow, care should be taken to insure that the arrowhead is placed near the point of entry to or exit from a connector. Connectors will be used if a line passing more than two symbols is required.

DECISION



This symbol represents a decision in the subroutine where alternate paths are possible. The accompanying text should include a statement of the comparison, decision, choice, or test. Each exit should be clearly identified.

PREDEFINED PROCESS



This symbol represents a group of operations which are not detailed in the flowchart. For example, a function or subroutine can be represented by this symbol. The name, if any, should be separated at the top of the symbol.

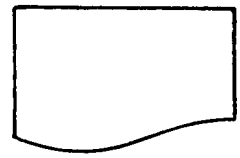
CONNECTOR

This symbol represents an entry to or exit from other sections of the flowchart.

CARD

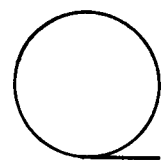
This symbol represents a punched card.

DOCUMENTS



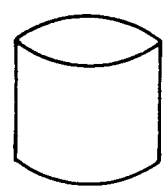
This symbol represents a computer printout.

MAGNETIC TAPE



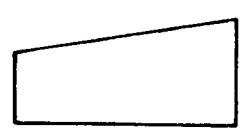
This symbol represents a magnetic tape.

DISK



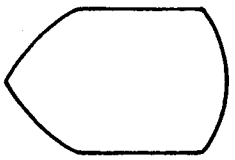
This symbol represents a disk.

DISPLAY KEYBOARD



This symbol represents a communication at execution time between the console or inquiry station and the computer.

DISPLAY

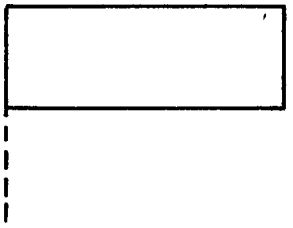


This symbol represents information which is displayed by a video device.

PAGE DELINEATOR

This symbol is used to indicate a break caused by the end of a page or line.

FLAG



This symbol represents a comment.

4.4 MANUALS

Task Specification, Subsystem Design, and User's Manuals will be written as the project is implemented. A task is defined as one or more subroutines and will be initiated by the Program Design Group completing a Task Specification. The individual Task Specifications will be the basis for the Task Specification Manual. The individual tasks will be assigned to a programmer. The programmer will complete the Subroutine Descriptions. The individual Subroutine Description will be the basis of the Subsystem Design Manual. A User's Manual will be written to enable the users of the system to select the necessary options. The User's Manual will be initiated at a later date.

The table of contents for the three manuals and the format for the Task Specifications, Subroutine Descriptions, and Cross-Reference Tables follow (Figures 4 to 9).

Table of Contents of Task Specification Manual

Section 1 - Introduction

Section 2 - General Flowchart of the System

Section 3 - Task Specifications

3.1
3.2
3.3
3.4
3.5
3.6
3.7
.
.
.

Section 4 - Glossary and Common Blocks

Section 5 - References

Figure 4. Table of Contents of Task Specification Manual

Table of Contents of Subsystem Design Manual

Figure 5. Table of Contents of Subsystem Design Manual

Table of Contents of User's Manual

Section 1 - Introduction

Section 2 - Set-up Procedures

Section 3 - Initialization Procedures

Section 4 - Hardware Requirements

Section 5 - Operator Intervention Requirements

Appendix - Sample Input and Output

Figure 6. Table of Contents of User's Manual

TASK SPECIFICATION FORMAT

Task Name _____

Contributor's Name _____

Date _____

PURPOSE

(This section will describe briefly the purpose of the task in the system. Reference will be made to the specific portion of the "mathematical requirements" which the task is designed to satisfy.)

METHOD

(This section will include the appropriate mathematical logic and formulae. It will also describe in a logical manner how the task is partitioned into convenient or logical segments.)

INTERFACE REQUIREMENTS

(All referenced storage areas external to this task will be described in this section. These storage areas include external tables and arrays, COMMON areas and/or Calling sequences.)

REQUIRED COMPLETION DATE _____

REFERENCES

(All applicable reference works will be tabulated in this section.)

Figure 7. Task Specification Format

SUBROUTINE DESCRIPTION FORMAT

Subroutine Name and Number _____

Programmer's Name _____

Date _____

PURPOSE

(This section will briefly describe the purpose of the subroutine in the task. Reference will be made to the particular portion of the task which the subroutine is designed to satisfy.)

METHOD

(This section will include the appropriate mathematical logic and formulae used in this subroutine.)

USAGE

(Specific descriptions of all referenced storage areas external to this subroutine will be given here, including the formats of external tables and arrays, COMMON areas, and calling sequences.)

CALLED SUBROUTINES

(This section will include a list of the subroutines which will be called by this subroutine along with the calling sequences.)

FLOWCHARTS

(If logic is involved in the subroutine, i.e., the subroutine does not merely compute a mathematical formula, a flowchart to the level of detail necessary to identify the computational flow and the interface between this subroutine and other subroutines is presented in this section. Flow is to be from left to right.)

RESTRICTIONS

(This section will list any restrictions.)

Figure 8. Subroutine Description Format

COMMON BLOCK CROSS-REFERENCE TABLE FORMAT

SUBROUTINE NAMES / COMMON BLOCK NAMES	SUBROUTINE 1								SUBROUTINE N
COMMON 1									
COMMON N									

SUBROUTINE CROSS-REFERENCE TABLE FORMAT

CALLING SUBROUTINES / SUBROUTINES CALLED	SUBROUTINE 1								SUBROUTINE N
SUBROUTINE 1									
SUBROUTINE N									

Figure 9. Cross-Reference Table Formats

4.5 PROJECT STANDARD FORMS

The following forms will be used during the development and implementation of computer programs for the project.

a. Weekly Progress Report (Figure 10)
b. Data Set Layout (Figure 11)
c. U.S. Government 2-Way Memo (Figure 12)
d. Project Change Request (Figure 13)
e. IBM 360 Computer System Problem Report (Figure 14)
f. Task Specification Assignment Form (Figure 15)
g. Task/Subroutine Assignment Form (Figure 16)
h. Parameter Name Glossary (Figure 17)
i. Subroutine Control List (Figure 18)
j. Documentation Control List (Figure 19)
k. Discrepancy Report Form (Figure 20)

WEEKLY PROGRESS REPORT

NAME: _____ WEEK ENDING _____

PROJECT _____

TASK _____ % COMPLETE _____ TIME TO COMPLETE _____

WORK PLANNED FOR THIS WEEK:

WORK COMPLETED THIS WEEK:

WORK PLANNED FOR NEXT WEEK:

COMMENTS, PROBLEMS, SUGGESTIONS ON THE PROJECT STATUS:

Figure 10. Weekly Progress Report

DATA SET LAYOUT

DEVICE CHARACTERISTICS

TAPE ☐ TAPE NO. _____

PARITY: EVEN (DEC) ☐ ODD (BIN) ☐
CHANNEL: 7 ☐ 9 ☐ OTHER _____
DENSITY: _____

DISK ☐ _____ DRUM ☐ _____ CELL ☐ _____
TRACK SIZE CYL SIZE

OTHER ☐ _____

_____

GENERAL

PHYSICAL UNIT _____ FORTRAN LOG UNIT _____
LOG RECORD SIZE _____ BLOCK SIZE _____
RECORD FORMAT _____

DSNAME: _____

WRITTEN BY: _____
(NAME OF SUBROUTINE)

READ BY: _____

PURPOSE: _____

_____

_____

_____

_____

_____

RECORD	RECORD SIZE	WORD	NAME, DESCRIPTION, ETC.

Figure 11. Data Set Layout

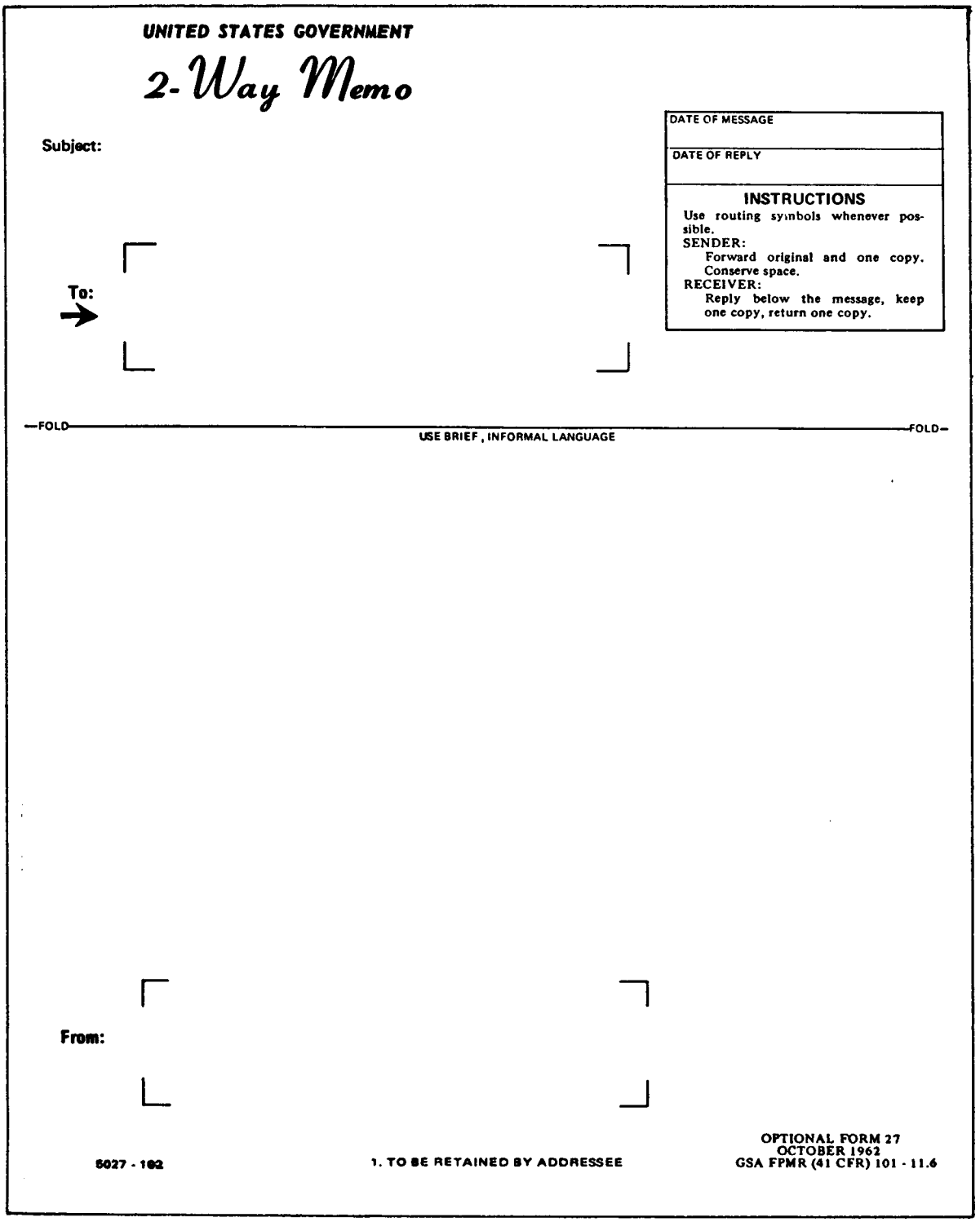


Figure 12. U.S. Government 2-Way Memo

PROJECT CHANGE REQUEST

TO: DATE: _____

FROM:

RE: MODIFICATION TO: _____

_____

DESCRIPTION OF CHANGE:

REASON FOR CHANGE:

PROGRAMS AFFECTED:

	ESTIMATED HOURS	PERSONNEL ASSIGNED	ESTIMATED COMPLETION DATE
PROGRAMMING DOCUMENTATION SYSTEM TESTING			

CHANGE INITIATED BY:	CHANGE APPROVED BY:

Figure 13. Project Change Request

IBM 360 COMPUTER SYSTEM PROBLEM REPORT

TO:

NAME _____ CURRENT DATE _____

LOCATION _____ DATE JOB RUN _____

PHONE _____ TIME JOB RUN _____

CODE/COMPANY _____ COMPUTER ASSIGNED JOB NUMBERS _____

MACHINE RUN ON _____

JOB NAME _____

1. AREA OF PROBLEM

____JCL ____I/O ____EXECUTION ____COMPILATION

2. YOUR DESCRIPTION OF THE PROBLEM (USE REVERSE SIDE IF NECESSARY).

3. TYPE OF PROGRAM ____RJE ____RITS/CRBE ____FORTRAN

____PLI ____ASSEMBLER ____GRAPHICS ____LINK EDIT ____UTILITY

____SORT/MERGE OTHER (SPECIFY) _____

4. HAS THIS (PRESENTED) PROGRAM EVER RUN SUCCESSFULLY? ____YES ____NO

5. HAVE ANY (EVEN ONE CARD) CHANGES BEEN MADE TO PROGRAM OR DATA? ____YES ____NO

IF SO, WHAT?

6. HAS PREVIOUS ASSISTANCE BEEN RENDERED FOR THIS PROGRAM PROBLEM? ____YES ____NO

7. MATERIAL BEING PROVIDED

____SOURCE LISTING ____OBJECT LISTING

____DECK ____TAPE NUMBER _____ _____

____DUMP - COMPLETION CODE _____

Figure 14. IBM 360 Computer System Problem Report

TASK SPECIFICATION ASSIGNMENT FORM

TASK NAME AND NUMBER	CONTRIBUTOR	START DATE	TARGET DATE	COMPLETION DATE	PROGRAMMER

Figure 15. Task Specification Assignment Form

TASK/SUBROUTINE ASSIGNMENT FORM

TASK NAME AND NUMBER	PROGRAMMER	START DATE	TARGET DATE	COMPLETION DATE

Figure 16. Task/Subroutine Assignment Form

PARAMETER NAME GLOSSARY

FORTRAN NAME	MATHEMATICAL SYMBOL	DEFINITION (from Standard Reference)

Figure 17. Parameter Name Glossary

SUBROUTINE CONTROL LIST

SUBROUTINE NAME	SUBROUTINE NUMBER	DOCU-MENTATION	LISTING	TEST	SOURCE DECK	OBJECT DECK	PROGRAMMER

Figure 18. Subroutine Control List

DOCUMENTATION CONTROL LIST

SUBROUTINE NAME	SUBROUTINE DOCUMENTATION	INITIAL EDIT	DRAFT TYPED	COMMENTS CORRECTIONS	FINAL TYPING	FINAL APPROVAL

Figure 19. Documentation Control List

DISCREPANCY REPORT FORM

DISCREPANCY REPORT

DATE: _____

OBSERVED ON TEST NO.: _____

OBSERVER: _____

ASSIGNED TO: _____

PROGRAM AFFECTED	DATA CASE INVOLVED
PROGRAM NAME _____	**WORKING FILES** _____
DECK ID _____	**PERMANENT FILES** _____

DESCRIPTION OF PROBLEM

(Completed by observer as soon as a discrepancy is recognized.)

Figure 20. Discrepancy Report Form

5. TESTING AND ACCEPTANCE TECHNIQUES

5.1 GENERAL

During the task, until acceptance by the task Technical Officer, the contractor will be required to maintain adequate backups of source programs, compiled programs, and test data sets. Source programs can be efficiently maintained on tape with the program SLIP (Ref. 4). This program allows for the selective updating of individual modules and the retention of different versions of the source program and data decks. The SLIP program is used on the IBM System 360, but most manufacturers supply similar utility programs as well as programs for preparing copies of other types of data sets. The member names should contain the date last modified. Only those subroutines still under development are to be compiled at test time. Those already checked out should be accessed from the subroutine library in the LINK step.

5.2 TESTING

Each subroutine will be tested individually by the author prior to integration with the whole program. The manager responsible for integration of the subroutine or function will require proof to substantiate the accuracy or correctness of each subroutine in the form of hand check and/or computer run comparison, as appropriate.

Subroutine testing should include all options and should use data generated by a driver written by the programmer. Test objectives and results should be indicated on hard-copy output and retained by the programmer until subroutine integration into a program structure has been completed successfully. Test data should include realistic numerical ranges. As subroutines are completed and tested, a master subroutine checklist should be updated by the project librarian. Items that are deliverable to the librarian include source/object decks, program listings and documentation, standard error code listings, and test reports and results.

On the program level, test cases should include all major options, as specified by the Functional Requirements, and as many additional options as feasible. Program level testing will be initiated by a Test Specification. After a test has been completed, Test and Discrepancy Reports, if any, will be completed. Test Specification and Test Report formats follow. The Discrepancy Report Form is shown in Section 4.

Subsystem and system tests will be of the parallel type, with comparison runs generated by existing systems and simulated data runs that will yield predictable results.

TEST SPECIFICATION FORMAT

TEST *(State objective of this particular test; i.e., what do you expect to prove by performing this experiment?)*

PROCEDURE

(This section will include data sources, internal options to be used, and output reports required.)

EVIDENCE

(Identify hard copy required to prove objective.)

QUANTITATIVE CRITERIA

(Describe items in reports to be compared with previously established numerical values. What are the values? If another job is required to establish these standards describe the job(s) or reference them. Prescribe acceptable tolerances. Show how measures are to be computed and how to determine whether the correct value was obtained in the test.)

Figure 21. Test Specification Format

TEST REPORT FORMAT

TEST REPORT **Test:** _____

Date: _____

Computer Run: _____

Analyst: _____

TEST

(Briefly describe the objective of this experiment. Note any differences in objective between this experiment and the corresponding experiment number described in the Test Specification.)

PROCEDURE

(Same as in Test Specification Format.)

EVIDENCE

(Include actual documentation obtained during the test. On generated reports, indicate which items were used and their purpose.)

QUANTITATIVE ANALYSIS

(Show how the quantitative criteria of the test specifications were met.)

QUALITATIVE ANALYSIS

(Does this experiment prove the objective? Conditionally? Are the exceptions significant or can they be ignored? Can the test be accepted after correcting the exceptions or should this test be rescheduled after the exceptions are corrected?)

Figure 22. Test Report Format

6. CORRECTION AND UPDATE STANDARDS

This section pertains primarily to large, multi-user systems. For smaller applications, the Technical Officer should perform the functions described below.

6.1 GENERAL

Standards to direct and control the modifications to the system must be established once the initial version of the system becomes operational. The objectives of these standards are to maintain one system which will respond to the requirements of the responsible organization and to eliminate the need for a proliferation of systems. Proper control will allow the system to remain dynamic and be made to reflect the state-of-the-art.

6.2 CONTROL COMMITTEE

6.2.1 Structure

A control committee with one or more members from the responsible organization, one member from the project, and one member from the computer center (Code 603), will have total responsibility and authority to make any changes necessary to maintain the status of the system. The members will be selected for a tenure of 12 months. Initially, two persons will serve for a 6-month period and two for a 12-month period. Thereafter, two new members will be selected every 6 months insuring continuity of the committee as well as providing valuable experience to various individuals in the respective branches. The members will select a chairman who will serve for a specified period and maintain a record of all committee actions.

6.2.2 Responsibility

The Control Committee will be responsible for evaluating and approving all proposed changes to the system. In the evaluation of a proposal, the committee will determine the impact on the system and the total cost (including manpower machine and documentation requirements). The committee will have the authority to obtain any assistance needed to properly evaluate any proposal and will be required to submit regular progress reports to project management. The proposer will be notified within 15 working days of the committee's action.

6.3 CHANGE PROCEDURE

6.3.1 Initiation of Proposal

All proposals should be submitted to the committee chairman using the U.S. Government 2-Way Memo (Optional Form 27, October 1962, GSA FPMR (41CFR) 101-11.6). Such proposals should have branch level approval. The respective branch head should initial the form prior to submittal.

Proposals will be classified as follows:

a. Problems and errors that cannot be resolved by the user
b. Corrections
c. Updates

All supporting documents and materials should be submitted with the proposal, including the Discrepancy Report Form for problems noted during operation.

After the committee has evaluated the proposal and submitted a reply, the proposer should indicate if the committee's action is acceptable or offer further suggestions. Upon failure of the committee to resolve the proposal, branch level resolution will be necessary.

6.3.2 Implementation of Change

In implementing a change, the committee chairman will submit a Project Change Request form to the group responsible for the maintenance of the system. He will also be responsible for:

a. Completion of the task
b. Proper testing of the modified system
c. Updating and distribution of documentation
d. Submitting notice, at least 30 days in advance, to all users indicating the change and the implementation date

The chairman will serve as the maintenance project leader, submit progress reports to the proposers, and maintain all necessary records. He is to have access to the necessary personnel and resources to perform this function.

The group responsible for updating the system should adhere to the requirements and standards as established in this document. The following materials should be submitted to the chairman of the Control Committee:

a. Completed Project Change Request form
b. Source deck with listing
c. Object deck with assembly listing
d. Test results
e. Corrected documentation to include:
    (1) Subroutine name
    (2) Programmer's name
    (3) Date of change
    (4) Variables affected, including COMMON BLOCKS
    (5) Modified flowchart

6.3.3 Documentation

All users are to be sent updated documents prior to the actual updating of the system. No change to the system will be considered complete until this has been completed. The system librarian will serve as the distributor of all documentation and notices to users.

7. OPERATIONAL GROUND RULES

As was mentioned previously, the SESD 360 computers are operated to provide rapid turnaround for the greatest number of users. To accomplish this, the priority of any job is inversely proportional to the resources it requires. Listed below are some of the ground rules that are currently in effect.

a. No block or dedicated time is given where other users are excluded.

b. No commitment of machine time by hours or runs is made to any individual or organization.

c. No user disk packs or data cells are permitted.

d. Priority is based upon use of system resources (CPU time, I/O time, core size, I/O devices). Smaller requirements get higher priorities.

e. Jobs are not permitted to have a dedicated printer, card reader, punch, or remote terminals.

f. Backup of disk data sets is the responsibility of the user.

g. Jobs requiring over 1 hour CPU or I/O time will have to be scheduled by the system manager.

h. No decimal arithmetic is permitted on the 360/91.

i. Jobs should not require operator communication via WTO/WTOR macros from application program.

j. No system modifications will be permitted. For instance a special SVG.

These rules have been established in order to make the SESD machines meet requirements for high throughput and fast turnaround. Additional rules may be established from time to time to help meet these objectives.

The contractor must keep these rules in mind in making estimates of elapsed time and computer time needed to complete this requirement. The programmer should note that the upper limit on job duration (sum of CPU and I/O time estimates) during prime shift is 6 minutes on the SESD 360/75 and 4 minutes on the SESD 360/91. A more realistic limit for quick turnaround is 1 to 2 minutes on the SESD 91 and 1 to 3 minutes on the SESD 75. With jobs in these limits, it is possible to expect the turnaround time to be most often about 1 to 2 hours.

The programmer should take care not to design into any program requirements for very large amounts of any resource. This specifically includes scratch disk space, core storage, and tape drives.

In addition to the above ground rules, the following information should be useful to programmers:

a. WTR NEWS - WTR NEWS supersedes all other information sources. It follows the header pages of every run. If a run has not been made for several days, a procedure is available to get back news.

b. NEWSLETTER - The GSFC Computer Newsletter is published every 2 weeks or as necessary. All programmers should be on the distribution list. A request can be made to receive all back copies. The Newsletter supersedes all other information sources except WTR NEWS.

c. The ground rules for machine usage as set forth above would appear at this level in the hierarchy of information.

d. The SESD Users' Guide - Each programmer should be furnished copies of the SESD Users' Guide. He should be instructed to become familiar with its contents.

For CRBE jobs the following restrictions apply:

a. Only 1- to 5- minute jobs (CPU + I/O), 300K maximum will be accepted to be run during the prime shift, 09:30 to 16:00 Monday through Friday.

b. No more than three jobs per I.D. of 1 to 5 minutes total time (CPU + I/O) in the job queue at one time.

c. After 16:00, 6- to 10-minute jobs will be accepted.

d. No more than two jobs of 6 to 8 minutes will be accepted, or no more than one job having a total estimated running time in the 9- to 14-minute range.

e. No job having an estimated running time of greater than 14 minutes will be accepted without prior approval from the Computer Manager.

f. During normal workload days, jobs other than special priorities over 300K and 5 minutes (CPU + I/O) will not be processed. After 17:00, jobs over 5 minutes will be processed in the order of their priority. Backlog is run by the earliest date.

REFERENCES

1. USAI Standard FORTRAN, American National Standards Institute, Inc., USA X3.9, 1966.

2. "Programming Standards Manual for the Goddard Trajectory Determination System," Goddard Space Flight Center, Contract Report NAS5-11723, August 1970.

3. "Computer Program Documentation Guideline," NASA Publication NHB 2411.1, July 1971.

4. "SLIP (Source Language Inquiry Program)," Space and Earth Sciences Computing Center User's Guide, Goddard Space Flight Center, May 1970, p. 240-241.

BIBLIOGRAPHY

1. Catalog of the GSFC Computer Program Library, Goddard Space Flight Center, recurring.

2. "Computer Program Documentation Guideline," NASA Publication NHB 2411.1, July 1971.

3 Dean, J. L., "Optimization Techniques for FORTRAN IV (G and H), Programs Written for the IBM 360 Under OS," Goddard Space Flight Center, X-543-71-99, March 1971.

4. GSFC Computer Newsletter, Goddard Space Flight Center, recurring.

5. M&DO IBM 360 Users Guide, Goddard Space Flight Center, September 1971.

6. "A Programmer's Guide to the Goddard Space Flight Center Computer Program Library," Goddard Space Flight Center, X-540-69-107, February 1969.

7. "Programming Standards Manual for the Goddard Trajectory Determination System," Goddard Space Flight Center, Contract Report NAS5-11723, August 1970.

8. "SLIP (Source Language Inquiry Program," Space and Earth Sciences Computing Center User's Guide, Goddard Space Flight Center, May 1970.

9. Space and Earth Sciences Computing Center User's Guide, Goddard Space Flight Center, recurring.

10. USAI Standard FORTRAN, American National Standards Institute, Inc., USA X3.9, 1966.